

Ministry of Higher Education
Herat University
Computer Science Faculty
Bachelor Thesis



**Comparative Performance Analysis of JavaFX and Windows Forms
(.NET Framework): A Case Study on Online Task Management
System**

Prepared by: Mohammad Yaser

Supervisor: Mr. Sayyed Kamran Hosseini

Year: 2024

رسالة محمد

In the name of the Almighty
Herat University
Computer Science Faculty
Software Engineering Department
Acknowledgment of Thesis Defense

It is hereby confirmed that the thesis of the honorable Mohammad Yaser son of Mir Wali Ahmad titled as Comparative Performance Analysis of JavaFX and Windows Forms (.NET Framework) A Case Study on Online Task Management System of under the supervision of the respected teacher, Seyyed Kamran Hosseini was written and the above-mentioned student has successfully defended his thesis on 2024/11/3.

1- Signature of the supervisor

2- Signature of the jury

3- Signature of the jury

4- Signature of the jury

With respect,
Abdul Khaleq Herawi
Head of Software Engineering Department

Abstract

Application systems performance immediately influences business metrics. Any form of production performance problem may lead to loss of revenue or customers. Among the most common frameworks, JavaFX and Windows Forms are popular choices. Usually, decision-makers are caught in a dilemma when it comes to the choice between the two, since there is a lack of practical studies comparing their performance. Most of the works just review the performance of their underlying languages, Java and C#, correspondingly, but none deeply compares the researched frameworks for real-world applications. The Aim of this thesis is to compare the performance of JavaFX and Windows Forms through a Task management system. A comparative experimental approach was followed in which the same application was developed with each framework. These applications were subjected to quantitative tests in respect of key performance metrics, which are the response time, memory usage, file operations, start-up time, CRUD operations to local and remote databases, and CPU usage. All performance data was then collected by the following: Visual VM, Performance Profiler, and Windows Performance Toolkit. The results also compared qualitatively with existing related literature about Java and C# performance. These tests showed that JavaFX outperformed Windows Forms in terms of memory consumption, remote CRUD operations, and response time. At the same time, Windows Forms outperformed JavaFX regarding startup time, local CRUD operations, CPU usage, and file operations. Based on such findings, a student can understand better where JavaFX and Windows Forms are strong and weak and also help them that which one of them is better to put their focus on it, this help developers and managers make appropriate choices concerning which framework to use for a particular project, also This work contributes to existing knowledge with regard to desktop frameworks and is also useful to be referred to by future researchers.

Keywords: JavaFX, Windows forms, Java, C#, Performance

Acknowledgment

On the personal level, I want to thank Allah most merciful for his blessings and guidance for having the privilege to work on this research. I am extremely grateful to my parents, who have always supported and encouraged me in every step of my educational journey. Their bountiful generosity and devotion have been instrumental in helping me take up this salient path, and it is because of them that I reach this milestone. I would like to express my deep gratitude to my respected mentor, Mr. Sayyed Kamran Hosseini, for his honest, wise, and unselfish guidance in all academic issues and the preparation of this thesis. His mentorship has been a real treasure during this journey. I respectfully and gratefully dedicate this work to these three brilliant supportive figures present in my life. The academic gift is like a "green leaf offered as a token of friendship," it is presented from the bottom of my heart.

Table of Contents

Chapter1 Introduction	1
1.1 Introduction	2
1.2 Problem Statement	3
1.3 Research Objective	3
1.4 Research Question	4
1.5 Research Significance	4
1.6 Research Hypothesis	5
1.7 Methodology.....	5
1.8 Research Structure.....	6
Chapter2 Literature Review	7
2.1 Introduction	8
2.2 Programming language	8
2.3 Java	8
2.4 JavaFx	9
2.5 C#	10
2.6 Microsoft's .NET	10
2.6.1 .NET Framework	11
2.6.1.1 Windows Forms	12
2.7 Literature review	13
Chapter3 Method & Materials.....	15
3.1 Introduction	16
3.2 Methodology.....	16
3.3 Materials	17
3.3.1 Performance Test Tools	17
3.3.1.1 Windows Performance Toolkit	17
3.3.1.2 VisualVm	17
3.3.1.3 Visual Studio Performance Profiler.....	17
3.3.2 Development Tools	17
3.3.2.1 Programming language and Framework.....	17
3.3.2.2 Integrated Dev8lopment Environment (IDE).....	18
3.3.2.3 Data Storage.....	18
3.3.3 Test Environment	18

3.3.3.1 Hardware	18
3.3.4 Case Study (Online Task Management System)	18
3.3.4.1 Development	18
3.3.4.2 Design	19
Chapter4 Result.....	25
4.1 Introduction	26
4.2 File operation	26
4.3 Response time	27
4.4 CRUD operation with MySQL	27
4.5 CRUD operation with Firebase	28
4.6 CPU usage	29
4.7 Memory Usage	30
4.8 Startup Time	31
Chapter5 Discussion	32
5.1 Introduction	33
5.2 File Operation	33
5.3 Respose time	33
5.4 CRUD Operation with Databases	34
5.5 CPU Usage	35
5.6 Memory Usage	35
5.7 Startup time	36
5.8 Summary	36
Chapter 6 Conclusion	37
6.1 Introduction	38
6.2 Conclusion	38
6.3 Limitation	39
6.4 Suggestion	40
References	42

Table of Figures

Figure 3.1: Windows Forms Login Page	19
Figure 3.2: JavaFx Login Page	19
Figure 3.3: Windows Forms Home Page	20
Figure 3.4: JavaFx Home Page	20
Figure 3.5: JavaFx User Management Page	21
Figure 3.6: Windows Forms User Management Page	21
Figure 3.7: Windows Forms Add User form	22
Figure 3.8: JavaFx Add User form	22
Figure 3.9: Windows forms Task creation page	23
Figure 3.10: JavaFx Task creation page	23
Figure 3.11: Windows Forms Task assignment page	24
Figure 3.12: JavaFx Task assignment page	24

Table of Tables

Table 4.1: JavaFx file operation test	26
Table 4.2: Windows Forms file operation test	26
Table 4.3: JavaFx Response time test	27
Table 4.4: Windows Forms Response time test	27
Table 4.5: JavaFx MySQL CRUD operation test	27
Table 4.6: Windows Foms MySQL CRUD operation test	28
Table 4.7: JavaFx Firebase CRUD operation test	28
Table 4.8: Windows Foms Firebase CRUD operation test	29
Table 4.9: JavaFx CPU usage test	29
Table 4.10: Windows Forms CPU usage test	30
Table 4.11: JavaFx Memory usage test.....	30
Table 4.12: Windows Forms Memory usage test	30
Table 4.13: JavaFx Startup time test	31
Table 4.14: Windows Forms Startup time test	31
Table 6.1 Conclusion Result summary	39

Chapter1

Introduction

1.1 Introduction

Applications on the desktop play a major role in people's lives, as this is how the system is directly connected with end users for the purpose of exchanging information and communicating. Currently, the place is full of demand for smart applications on the desktop which are capable of meeting users' needs.

Application systems performance plays a huge role in business success, since any performance related issues in production may lead to a loss of money and potentially drive away customers (Heger et al., 2017). By far, some of the most popular frameworks for building such applications are JavaFX and Windows Forms (.NET Framework).

JavaFX is a family of next-generation graphics and media packages that allows developers to easily design, test, and deploy rich client applications across browsers, desktops, mobile devices, televisions, and other devices consistently (JavaFX Overview, n.d). In other words, JavaFX is a Java library applied to the development of RIAs. An application built using this tool will run on a wide array of hardware devices, from desktops down to mobile phones and even on television and tablets. Before the emergence of JavaFX, developers relied mainly upon the Advanced Window Toolkit, Applets, and Swing for developing GUI applications. JavaFX facilitated easier construction of GUI apps; as such, Java programmers built content-rich applications with greater efficiency than in the past (Cheshta & Sharma, 2017).

Windows Forms contradicts the .NET Framework technology for developing smart-client applications by making the development of certain tasks, such as file system operations, pretty simple. It allows the application developer to create an application that can display information and gather user input, and connect to remote computers over a network. Development environments like Visual Studio make it more easy to create Windows Forms applications to deal with these kinds of tasks (Adegeo, 2023).

The objective that this thesis seeks to achieve is to benchmark two frameworks, one is JavaFX and the second is Windows Forms (.NET Framework), in the course of developing a task management system which is able to function in both online and offline modes for a computer science faculty. Such discrepancy focuses on numerous performance metrics, including application's response time during instances of user interaction such as hitting a button, app's loading time during app's initiation, memory usage, file operations performance such as reading and writing documents, performing CRUD actions against local and remote (web)

databases and even CPU utilization. Finally, this research will seek to contrast all the above mentioned performance metrics with results of previous studies so as to place this analysis in larger perspective. The goal of the thesis is to make recommendations to decision makers regarding the selection of the most optimal library based on performance and speed while boosting the quality of applications.

1.2 Problem Statement

In order to develop desktop applications, it is important to select the right frameworks that provide good performance and usability. When one selects a proper framework for their application, they are bound to generate an application that is quick, responsive, and utilizes system resources well in order to ensure great user experience.

On the other hand, Usually, decision-makers are caught in a dilemma when it comes to the choice between the two, since there is a lack of practical studies comparing their performance.

Thus, if the decision makers choose a framework badly, they may face serious challenges. These can relate to lower performance, less effective resource utilization, longer cycles, and technical problems that could hurt the quality of their applications. Without proper and clear comprehension of the performance of both, a JavaFX and Windows Forms developer will make decisions which are detrimental to the success of the projects. It's thus important that thorough comparison of the two frameworks be done to help decision makers make better choices.

1.3 Research Objective

1.3.1 Main Objective

- To Compare the performance of JavaFX and Windows forms (.Net framework) on Online Task Management system

1.3.2 Sub-Objectives

- To measure the performance of file operations in JavaFX and Windows Forms application.
- To measure the response time for user interactions in JavaFX and Windows Forms application.
- To measure the efficiency of CRUD operations in JavaFX and Windows Forms application.
- To evaluate the CPU usage of JavaFX and Windows Forms application.

- To analyze the memory usage of JavaFX and Windows Forms application.
- To determine the application startup time in JavaFX and Windows Forms application.

1.4 Research Questions:

1.4.1 Main Question:

- What are the performance differences between JavaFX and Windows Forms (.NET Framework) in the online task management system?

1.4.2 Sub-Questions:

- What are the differences in file operation performance between JavaFX and Windows Forms applications?
- What are the differences in response times for user interactions between JavaFX and Windows Forms applications?
- What are the differences in the CRUD operations between JavaFX and Windows Forms applications when interacting with databases?
- What are the differences in CPU usage between JavaFX and Windows Forms applications?
- What are the differences in memory usage between JavaFX and Windows Forms applications?
- What are the differences in application startup times between JavaFX and Windows Forms applications?

1.5 Research Significance

The present study compares the performance of JavaFX and windows forms in the development of desktop applications. Here is The significance of the research from the perspective of different stakeholders.

For Students: This could be very helpful to understand the pros and cons of both JavaFX and Windows Forms. It would help students to choose which technology to concentrate on, according to their interests, as well as according to the emerging demand in the job market for any of the two frameworks.

For Developers: It will also be useful to developers when they have to choose a framework for their next project, because this paper discusses about performance, efficiency, and usability

of JavaFX and Windows Forms. This makes it easier for the developer to make a more informed decision which suits the best for develop their project.

For Managers: The project managers or technical leads take this analysis for deciding to choose which framework is best for their project. It would help them to understand the long-term effect of their choice, in terms of speed of development and resource allocation, on overall project costs and see that the chosen frameworks are aligned with business goals.

For Researchers: This is a study involving JavaFX and Windows Forms, which will enable them to compare their work with the one presented here, or for increase their understanding about JavaFx and windows forms.

1.6 Research Hypothesis

The hypothesis of this present thesis is that significant differences will be found in JavaFX or Windows Forms performance for developing desktop applications. It is expected that, during this research, differences in performance between JavaFX and Windows Forms will emerge in relation to user interaction. Supposedly, JavaFX will respond faster, while Windows Forms may be better in terms of CPU performance. It is also foreseen that Windows Forms will be more effective in performing CRUD operations, whereas JavaFX will show less consumption of memory during its processes. Besides, it is expected that JavaFX will be faster in starting compared to Windows Forms, and the Windows Form will outperform JavaFX in its file operation handling. These hypotheses can be proven or nullified through appropriate practical tests.

1.7 Research Methodology

This thesis aims to compare the performances of two frameworks, JavaFx and Windows Forms. Regarding the goal, for each of the frameworks, an Online Task management system application is developed. Also, in this research work, a combination of qualitative and quantitative research methods has been used. In this work, quantitative methods are used for collecting statistical data regarding performance metrics, whereas qualitative methods have been employed to analyze and compare from previous studies

1.8 Research Structure

Chapter 1: Introduction: Overview and introduction to general topics.

Chapter 2: Literature Review: Review of previous studies relation to JavaFx and Windows forms.

Chapter 3: Methodology: Upon what methods and material are this thesis performed.

Chapter 4: Results: Tables related to the JavaFx and windows forms.

Chapter 5: Discussion: Interpretation the results between JavaFx and Windows form.

Chapter 6: Conclusion: Summary of the key findings the performance of JavaFx and Windows forms.

Chapter 2

Literature Review

2.1 Introduction

Performance issues of programming languages have been studied extensively, and obviously, that has huge impacts on the performance of software. Since that topic is so rare, there are no certain articles that make direct comparisons between performance in Windows Forms with JavaFX. However, several reputable articles provide insight into the performance of C# and Java. These languages are the bases of these frameworks; hence, an attempt is made in this chapter to review most of the relevant research with regard to this research, as well as to provide an overview of the two frameworks and their base languages.

2.2 Programming language

A programming language is much like any other language in that it is a means of using instructions that a computer can execute. It includes syntax, which defines the form of code; semantics, while giving meaning to the code; a standard library available for most programming languages, useful in utility functions for common tasks; and an ecosystem-inclusive community support and third-party libraries that extend functionality (Hong & Ryu, 2023).

2.3 Java

The microprocessor revolution greatly facilitated personal computers and made a lasting impact on smart consumer-electronic devices. Realizing this trend, Sun Microsystems in 1991 launched an internal research effort led by James Gosling that resulted in Java—a C++-based object-oriented programming language. From its outset, one of the fundamental goals in designing Java was to make it possible for programs to execute on as many different types of computer systems and devices as possible; this is sometimes summarized as "write once, run anywhere." Shortly after Oracle's acquisition of Sun in 2009, the company announced at the JavaOne 2010 conference that Java ran on 97% of enterprise desktops, three billion mobile devices, and 80 million TVs. To date, more Java programmers are emerging, boasting the highest number of over 9 million developers, hence making it the most used programming language in the world today (Deitel, H. & Deitel, P., 2013).

Java has also provided the base for some of the most famous frameworks which leverage the efficiency of application development. The most popular java frameworks include: Spring, Spring Boot, Hibernate, JavaServer Faces, Struts, Grails, Vaadin, and JavaFX. While these

frameworks have different purposes, this thesis will narrow its scope to JavaFX and its function in modern desktop application development.

2.4 JavaFX

JavaFX 2 is a significant milestone in the evolution of Java into a rich client platform. As such, it is specifically created to be as lightweight as possible and hardware-accelerated for UI functionality when creating enterprise and business applications. JavaFX 2.0 was launched in October 2011, even though the first version was launched in the year 2007. One of the key benefits is that, as part of JavaFX 2.0, a developer can code in Java and make use of mature and widely-used development tools (Laying out a user interface with JavaFX 2.0, n.d.). JavaFX is generally a Java-based library which is implemented to create RIAs. Applications developed in JavaFX are planned in such a way that they function smoothly for desktops, mobile devices, TVs, and tablets. Until JavaFX, developers used various tools such as AWT, Applets, and Swing in order to build GUI applications. However, JavaFX makes development easier as it enables the Java programmer to build the rich content GUI applications with much efficiency. Earlier, developers had different libraries for embedding media handling, UI controls, web functionalities, and both 2D and 3D graphics into their client-side applications. JavaFX, however, incorporates all these into one extensive library. Moreover, all the other tools that exist in standard Java such as Swing and Applets—are still accessible by the developer in conjunction with JavaFX. It will also have strong graphics and media API support that would allow the utilization of the latest Graphics Processing Units through hardware acceleration for graphics processing (Cheshta & Sharma, 2017).

Key Features of JavaFX:

- **Built in Java:** The JavaFX library is built on pure Java and supports all JVM-based languages including, but not limited to, Java, Groovy, and JRuby. Because the application with JavaFX is developed based on Java, the application is automatically platform-independent.
- **FXML:** JavaFX arrives with a new declarative language, called FXML, which is similar to HTML UI.
- **Scene Builder:** This enables developers to visually construct user interfaces, by dragging and dropping elements. It can work with IDEs like NetBeans to reduce the hassle of UI creation.

- **Rich UI Controls:** JavaFX offers a range of controls that a developer can use to build comprehensive applications.
- **CSS-like Styling:** With JavaFX, developers can apply CSS-like styling to enhance the appearance of applications.
- **Integrated Graphics Library:** JavaFX provides support for 2D and 3D graphics through the Prism graphics pipeline. The above pipeline is hardware-accelerated. Hence, it runs quite smoothly with the help of supported graphics cards or GPUs. If this is unsupported by hardware, then software rendering of JavaFX will be applied.

2.5 C#

The C# programming language was created by Microsoft in the year 2000. It is based on ideas borrowed from such languages as C, C++, and Java. C# is similar to Java; both possess similar functionalities that have enabled them to be utilized in some of the most complex and challenging development projects. These include enterprise-level applications up to web, mobile, and cloud-based applications (Deitel, H., & Deitel, P., 2016).

2.6 Microsoft's .NET

Also in 2000, Microsoft announced its .NET initiative, which is an all-encompassing road map for developing, engineering, and delivering software using the internet and the web. One of the hallmarks of .NET is that it supports multiple programming languages, allowing developers to create applications in any language that targets .NET, including, but not limited to, C#, Visual Basic, and Visual C++. A crucial component of .NET is ASP.NET, which powers the development of dynamic web applications (Deitel, H., & Deitel, P., 2016).

.NET is a free, cross-platform, open-source framework designed to support the development of a variety of application types. C# is the dominant language within this ecosystem. The platform is equipped with a high-performance runtime, delivering enhanced productivity, security, performance, and reliability. The runtime itself automatically manages memory through its garbage collector (GC), thus allowing for both type safety and memory safety. Concurrency is achievable with the help of asynchronous programming and primitives like Task. Another advantage of the platform is that a huge amount of libraries—managed under the functionality coverage and optimized for many operating systems and hardware architectures—can be used by developers. Centrally, the .NET architecture has methods, libraries, and languages that provide a foundation upon which higher-level tools and

components, such as ASP.NET Core, are built. C# is the primary language for the .NET framework in which most of the platform itself is maintained in C# (Gewarren, 2024).

.NET ecosystem: There are several variants of the .NET ecosystem that are designed for different types of applications. These variants are partly due to historical reasons, but also for some technical consideration.

.NET Implementation: There are several implementations of .NET:

- **.NET Framework:** This is the original version of .NET that offers access to the broad functionalities of Windows as well as Windows Server. It is still actively supported, though mainly in a support capacity.
- **Mono:** The original community-driven or open-source version of .NET. This is the cross-platform implementation of .NET Framework that is being actively maintained for Android, iOS, and WebAssembly.
- **.NET Core:** Also commonly referred to as modern .NET, this is a cross-platform open-source implementation that was totally re-engineered from the ground up with the cloud era in mind. It remains very compatible with the original .NET Framework. It's being actively supported for Linux, macOS, and Windows.

2.6.1 .NET Framework

The .NET Framework provides the runtime environment that executes applications and the .NET Framework Class Library, which contains the classes, interfaces, delegates, etc., that provide a high degree of functionality for C# programs to perform complex tasks. The Class Library contains an enormous number of pre-built classes, which are high-performance tested and optimized. As you learn to develop your own classes, you are encouraged to reuse pre-existing classes from .NET Framework whenever possible. This generally saves you development time and can often improve the quality and efficiency of your software products (Deitel, H., & Deitel, P., 2016).

NET Framework Class Library is a part of the reusable types in the .NET Framework tightly integrated with the common language runtime. These types are object-oriented, allowing your managed code to inherit functionality, which makes them easier to use and quick to learn from the various features of the framework. In addition, third-party components easily integrate with classes from the .NET Framework. You can use the .NET Framework to develop many types of applications and services, including: Console applications, Windows GUI applications,

using a subset of the .NET Framework called Windows Forms Windows Presentation Foundation (WPF) applications, ASP.NET applications, Windows services, Service-oriented applications, using WCF, Workflow-enabled applications, using WF. The classes for Windows Forms represent an extensive library of reusable types that dramatically simplify Windows GUI development. Similarly, if you create an ASP.NET Web Form application, you can utilize the Web Forms classes (Gewarren, 2023).

2.6.1.1 Windows Forms

WinForms was first released along with the original version of the .NET and Visual Studio in 2001. It can be thought of as an abstraction over the very complicated Win32 API. It also provided the corporate developer a way to create data-bound line-of-business applications without having to be a guru in C++. WinForms gained popularity overnight due to its WYSIWYG designer—even an inexperienced developer could compose an application in minutes, satisfying some business needs (Loeffelmann, 2022).

Windows Forms is the smart client technology for the .NET Framework and consists of managed libraries that make it easier to carry out common activities, such as reading from and writing to the file system. Using a development environment such as Visual Studio, programmers can create Windows Forms smart-client applications that have graphical representations of data input from or output to the user of the program and communicate with other computers over a network. Generally speaking, Windows Forms is comprised of a form, which is a graphical surface for displaying information to the user. Programmers typically write Windows Forms applications by dragging controls onto forms and then writing code to respond to events associated with the controls, such as mouse clicks or key presses. A control is a discrete UI element that displays data or provides input for data. Windows Forms contains a set of controls the developer can place on forms, such as text boxes, buttons, drop-down boxes, and radio buttons. Even web pages can be placed in a Windows application. If the provided controls do not meet the needs of the application, the developer can create a custom control by inheriting from the User Control class. Windows Forms contains a plethora of UI controls that mimic many of the high-end application components found in Microsoft Office products. With the Tool Strip and Menu Strip controls, for example, it's easy to build toolbars and menus that can display text, images, and even submenus, and host other controls, such as text boxes and combo boxes. With the drag-and-drop Windows Forms Designer in Visual Studio, development is fairly drag-and-drop easy. The designer even allows developers to choose

controls and drop them onto a form with their cursor. Various tools, such as gridlines and snap lines, allow the designer to align controls from within the designer. Advanced form layouts are almost trivial, whether using the forms designer in Visual Studio or compiling from the command line using controls like `FlowLayoutPanel`, `TableLayoutPanel`, and `SplitContainer` (Adegeo, 2023).

2.7 Literature review

Such aspects as execution time of algorithms, speed of operations, code modification flexibility, and performance in general were reviewed and tested by the authors of “Java Programming Languages: Time Permanence with Other Languages”. The results showed that, on average, Java performed better, especially in operations with big datasets, while smaller sets were better handled by C++. Java was considerably slower than C# on parallel processing processes, especially in evolutionary algorithms such as SOMA. However, it was also much faster running than Python because of compilation and good garbage collection for larger-sized data. Java also turned out to be faster compared to C when executing matrix multiplication (Hamaamin et al., 2024).

Comparative Analysis of C, Fortran, C#, and Java Programming Languages by Biswa et al. (2016). Run several case tests, such as finding out the mean of numbers, bubble sort algorithm, reading/displaying files, and a benchmark application. The results of the tests indicated that Java performed better in general execution time in sorting and mean calculations; the best in the benchmark test was C. C# was more efficient in file operations. Fortran was the slowest of all the above tests. It is concluded that object-oriented languages, like Java and C#, are more efficient compared to procedural languages like C and Fortran (Biswa et al., 2016).

Chen compared C, C++, C#, and Java in Five Operational Steps: int arithmetic, double arithmetic, long arithmetic, trigonometric calculation (sin, cos, tan, log, square root), and file I/O. The results showed that in all the five operations, C was the third best while C++ was the second best. The maximum speed of C# was for int, double, trig arithmetic, and I/O file operation but the minimum speed for long arithmetic. Surprisingly, Java was the best for int, double, and long arithmetic but it was the poorest in trig arithmetic and I/O file operation. In summation, Java was ranked the last in total performance, which spent far more time compared with the other three languages (Chen, 2010).

In “Performance Comparison of Java and C++ When Sorting Integers and Reading/Writing Files” the author investigated strengths and weaknesses of C++ and Java in three most extensively used fields: data loading, sorting and saving. Sharma worked out two applications for each language which loaded three files containing 1,000, 10,000, and 100,000 randomly ordered integers. The results showed that Java was, overall, faster regarding the total time taken and that, generally, it was quicker to load two of the three datasets. It is also possible to notice that C++ was generally faster during the sorting of data with quicksort and insertion sort, as well as saving data to files. It was observed that Java felt overall faster, while C++ handled the heavy duty better (Sharma, 2019).

Pajjuri et al. (2001) discussed the results of a performance comparison of Java versus C by running eight algorithms, namely FIR, Matrix multiplication, Message digest, Acker's function, Fibonacci series, Simple hash, Array copy, and String concatenation, on both operating systems Linux and Windows with respect to CPU efficiency and memory consumption. The result indicated that, in the case of most CPU performances, which included FIR, MD5, matrix operations, and string concatenation, C was better; Java outperformed in the Fibonacci series. In fact, C outperforms Java in the case of memory utilization in DSP operations as regards FIR, MD5, matrix operations, and hash functions, and Java outperforms C in Fibonacci. Hence, C was generally better than java (Pajjuri et al., 2001).

Chapter 3

Method & Materials

3.1 Introduction

The present thesis compares the two Frameworks: JavaFx and Windows forms. To carry out this purpose, an online Task management system application is developed for each framework beside a combination of qualitative and quantitative research methods. The combination of these methods will, therefore, enable us to comprehensively and precisely evaluate these two frameworks and precisely identify the strengths and weaknesses of each. Quantitative methods have been used to collect statistical data on performance metrics, while qualitative methods were used for analyzing and comparing the findings with previous studies. The details of each of these methods and how to apply them in this research will be given in detail below.

3.2 Methodology

Quantitative strategy in this research Additionally, it measures the performance functions of these applications in controlled test settings. Some key performance indicators will include response time for interactive user operations like clicking buttons, application startup time, memory consumption during runtime, file manipulation efficiency, execution speeds of CRUD operations, local and remote databases, and CPU usage in various application procedures. Performance data is measured precisely by the tools Visual VM and Performance Profiler—both in Visual Studio—and Windows Performance Toolkit. These utilities will provide the capability to collect detailed performance measures that allow for real evaluation and comparison of the frameworks.

The qualitative part of this thesis involves the comparison of the findings presented in the performance evaluation between JavaFX and Windows Forms against any previously established results set by other related studies. This will help in advancing the current findings within the context of prior writings on desktop application frameworks. Through the analysis of similarities or discrepancies from prior research, insight into the strengths and weaknesses of each framework can be derived. Therefore, such a combination of quantitative and qualitative methods contributes to an inclusive assessment of JavaFX and Windows Forms' performances in view of the task management system.

3.3 Materials

3.3.1 Performance Test Tools:

3.3.1.1 Windows Performance Toolkit

The Windows Performance Toolkit is a set of tools used for monitoring performance; it produces in-depth performance profiles of Windows operating systems and applications. The Windows Performance Toolkit includes two independent tools: Windows Performance Recorder (WPR) and Windows Performance Analyzer. All recordings will need to be opened and analyzed by means of WPA.

3.3.1.2 VisualVM Testing Tool

Java VisualVM is a visual tool that provides detailed information about Java applications running on a Java Virtual Machine, or JVM. Java application developers can use it for troubleshooting applications and also monitor and improve the performance of applications.

3.3.1.3 Visual Studio Performance Profiler

Visual Studio contains a suite of profiling and diagnostics tools that can help you diagnose any possible memory or CPU usage and other application-level issues. These tools allow you to accumulate performance data as you run your application. A profiler can enable you to make informed decisions quickly with which execution times and CPU usage may be visually depicted for your application.

3.3.2 Development Tools

3.3.2.1 Programming language and Framework

- **Java:** Java is also famous for its cross-operability: a program made in Java can work on a computer with an arbitrary OS if it has JVM. Thus, it is well utilized in web, mobile, and desktop technologies, and especially goes to enterprise development and Android development-shine.

JDK Version: 21.0.1

- **C#:** C# is a language developed by Microsoft, and it's mainly applied to Windows applications, although with the arrival of .NET Core, it became more cross-platform. It is noted for its strong integration with the Windows ecosystem and generally applied in desktop, web, and game application development.

- **JavaFx:** JavaFX is a modern Java framework that is used to build rich, winning GUI applications. Great for cross-operating systems, it can enable developers to build feature-rich cross-platform applications with various great features such as animation, styling with CSS, responsive layouts, and much more.

Version: 21-ea+24

- **Windows Forms:** Windows Forms was a .NET framework from older times that was used to create applications targeted only at Windows. It is simple to use and has a huge library of controls, thus being ideal for rapid development of simple applications based on Windows.

.NET Framework Version: 4.8

3.3.2.2 Integrated Development Environment (IDE)

- IntelliJ IDEA for JavaFx
- Visual Studio for Windows forms

3.3.2.3 Data storage

- MySQL Version 8.0.36
- Firebase Version 9.4.1

3.3.3 Test Environment

3.3.3.1 Hardware

- **Device Name:** DESKTOP-UAGQMOQ
- **Processor:** Intel Core i7-4810MQ CPU @ 2.80 GHz
- **Installed RAM:** 16.0 GB
- **System Type:** 64-bit operating system, x64-based processor

3.3.4 Case Study (Online Task Management system)

3.3.4.1 Development

Computer Science Task Management System, developed for the purpose of this study, is a multi-user application created for a different role played within the university setting: the Dean,

the Head, the Committee Members, and the Department Members. Each respective role has different functions and rules with an interface that uniquely derives their responsibilities. It works like a tree, where higher-order users—for example, the Dean—can create and assign tasks to their subordinates, such as Heads and Committee Members, who would further create tasks and assign them to others. Department Members mainly report tasks to the Dean. The system supports features such as notifications and real-time updates for smooth task management, allowing for improved workflow across all roles.

3.3.4.2 Design

In this section it will be nice that I show you the most important parts of the app the we will have deal with it during application tests.

- **Login page:** The login page is where that those users that are parts of a faculty can enter that. Other users that they are not part of a faculty then cannot enter to app even if they have this app. Figure 3.1 and Figure 3.2 shows the Login page for both frameworks.



Figure 3.1 Windows Forms Login Page

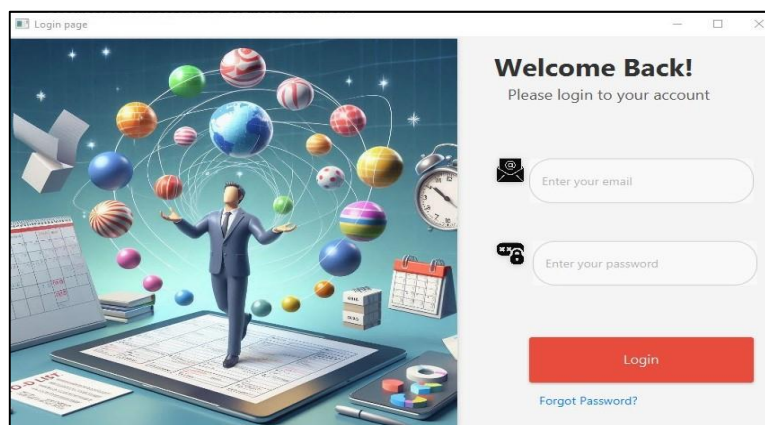


Figure 3.1 JavaFx Login Page

- **Home page:** This is the initial page while the app start. Figure 3.3 and Figure 3.4 shows the Home page for both frameworks.

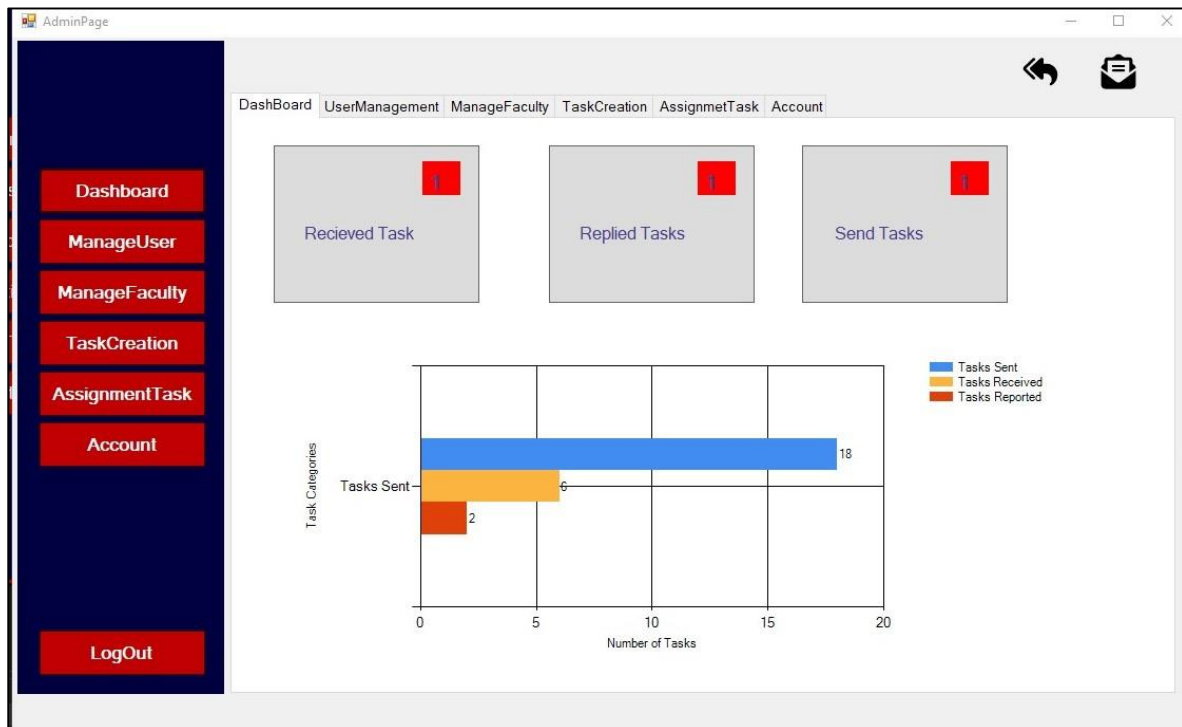


Figure 2.3 Windows Forms Home Page

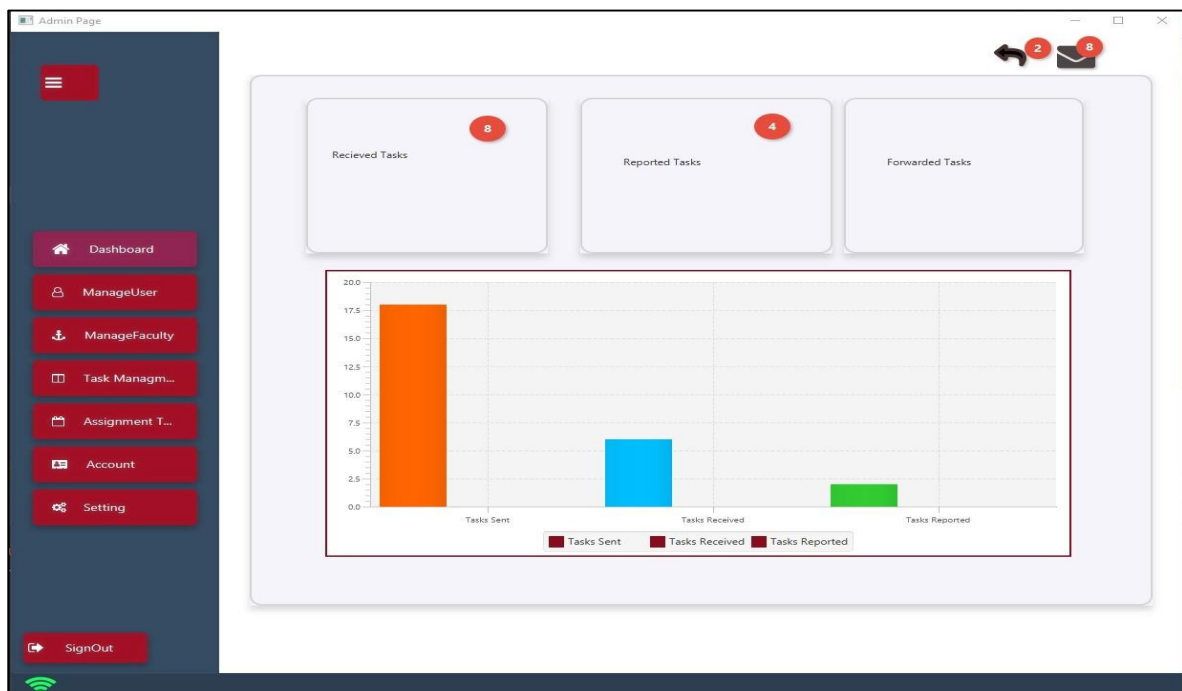


Figure 3.3 JavaFx Home page

- **User Management page:** This is where that can admin performs CRUD operations on user. Also is this page section we perform CRUD operation Test for this thesis. Figure 3.5 and Figure 3.6 shows the Page for both frameworks app.

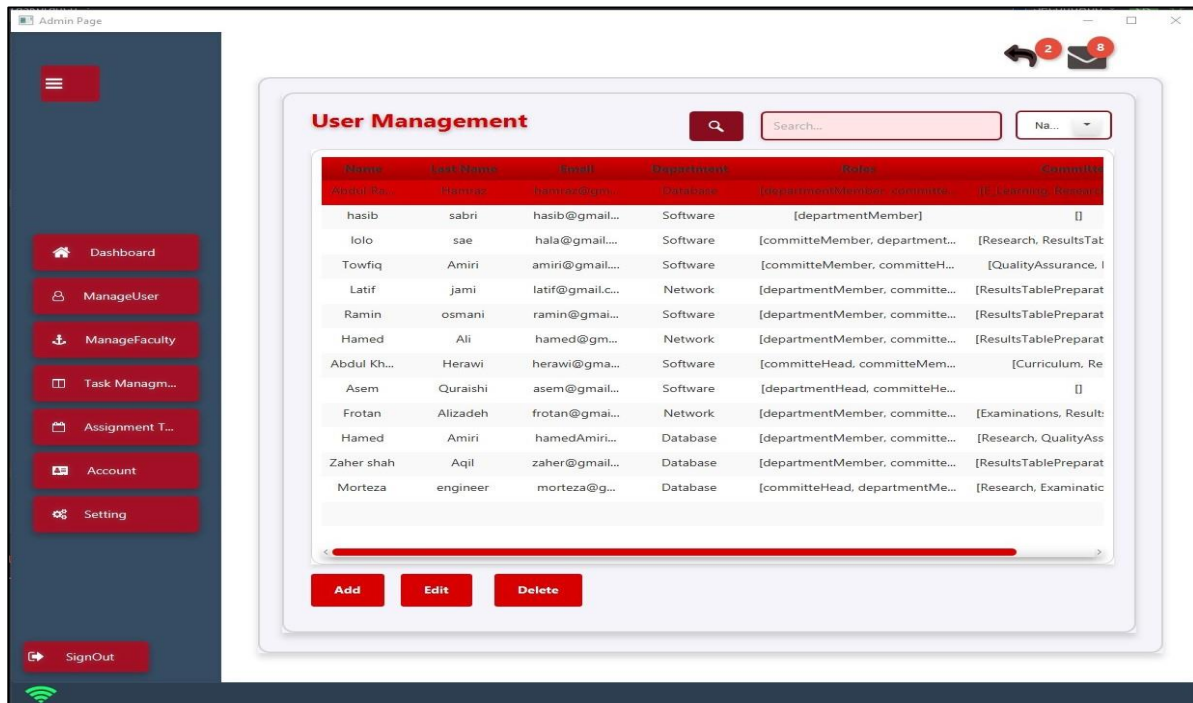


Figure 3.4 JavaFx User Management page

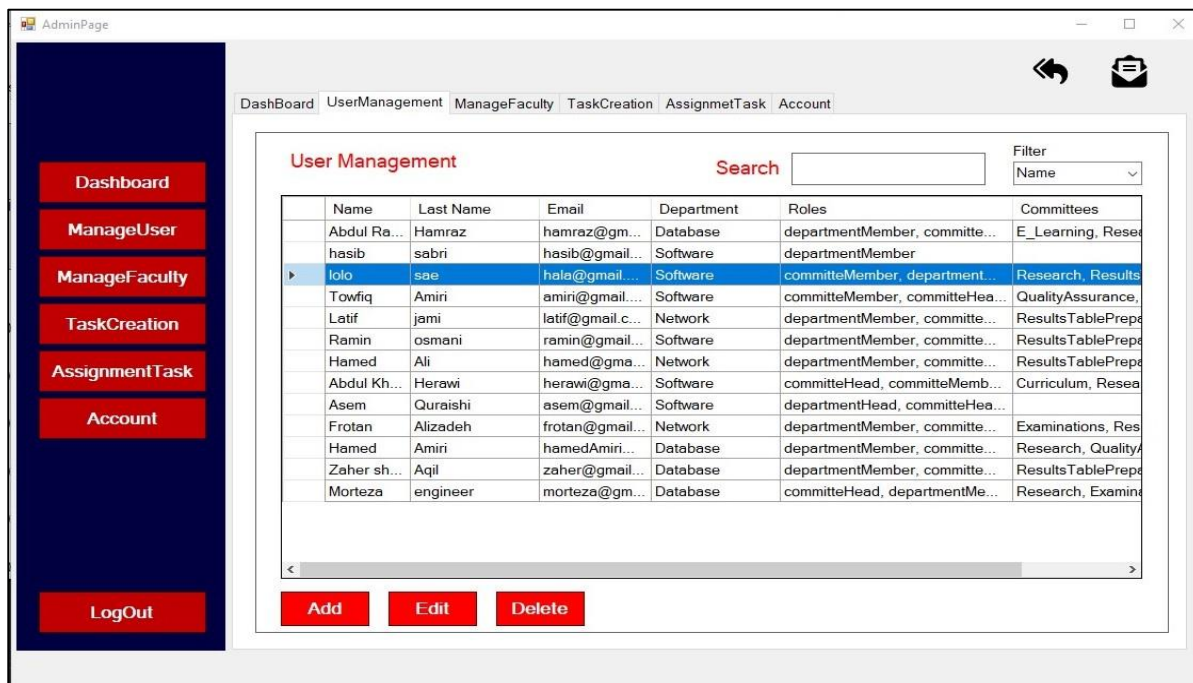
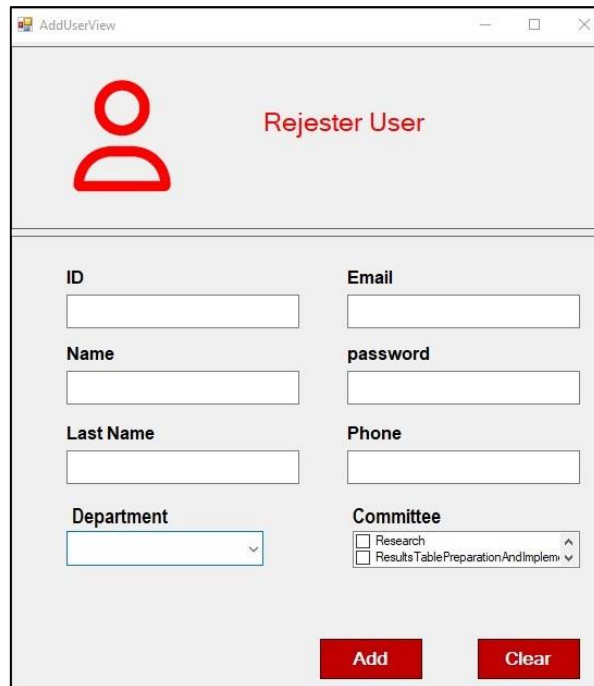


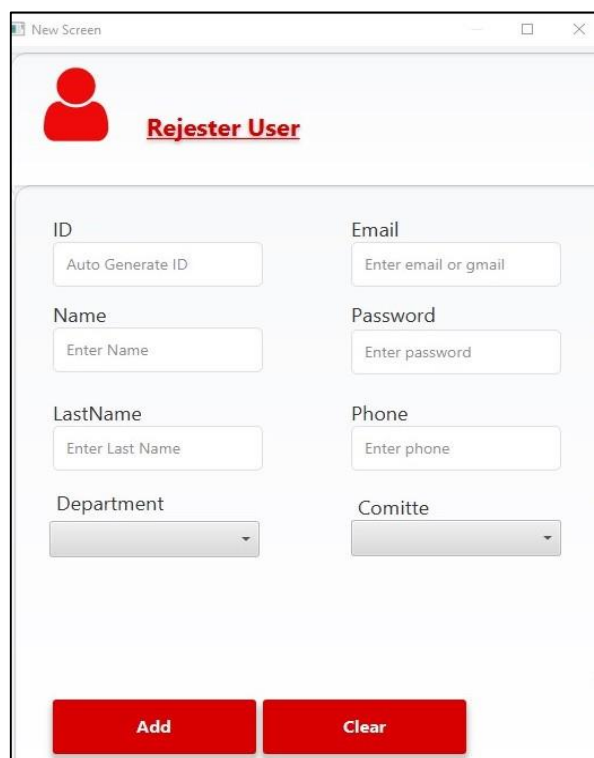
Figure 3.5 Windows forms User Management page

- **Add User form:** This is where that we can add User to system. This action can only be done by Dean. Figure 3.7and Figure 3.8 shows the form for both frame works



The screenshot shows a window titled "AddUserView". At the top, there is a red icon of a person and the text "Rejester User". Below this, the form is organized into two columns. The left column contains fields for "ID", "Name", "Last Name", and "Department" (a dropdown menu). The right column contains fields for "Email", "password", "Phone", and "Committee" (a dropdown menu with options "Research" and "Results TablePreparationAndImplem"). At the bottom right, there are two red buttons labeled "Add" and "Clear".

Figure 3.6 Windows forms Add user form



The screenshot shows a window titled "New Screen". At the top, there is a red icon of a person and the text "Rejester User". Below this, the form is organized into two columns. The left column contains fields for "ID" (with a button "Auto Generate ID"), "Name" (with placeholder "Enter Name"), "LastName" (with placeholder "Enter Last Name"), and "Department" (a dropdown menu). The right column contains fields for "Email" (with placeholder "Enter email or gmail"), "Password" (with placeholder "Enter password"), "Phone" (with placeholder "Enter phone"), and "Comitte" (a dropdown menu). At the bottom, there are two red buttons labeled "Add" and "Clear".

Figure 3.7 JavaFx Add user form

- **Task Creation page:** This is where that each user can create custom tasks that later can assign them to other users; also in here we will perform user interaction response time test while user click on create button. Figure 3.9 and Figure 3.10 shows the page for both frameworks.

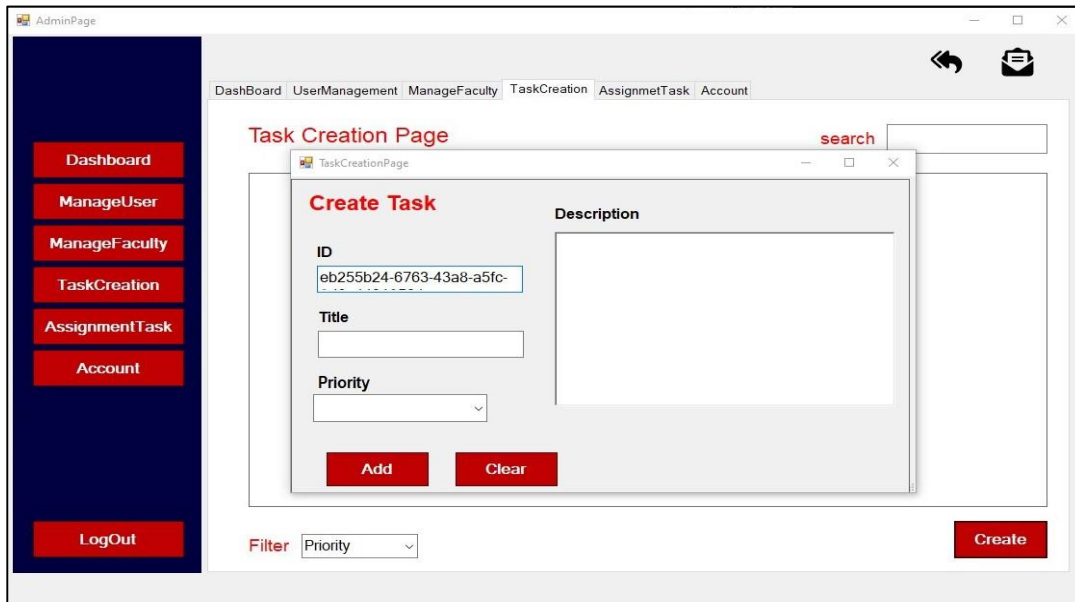


Figure 3.8 windows forms Task creation page

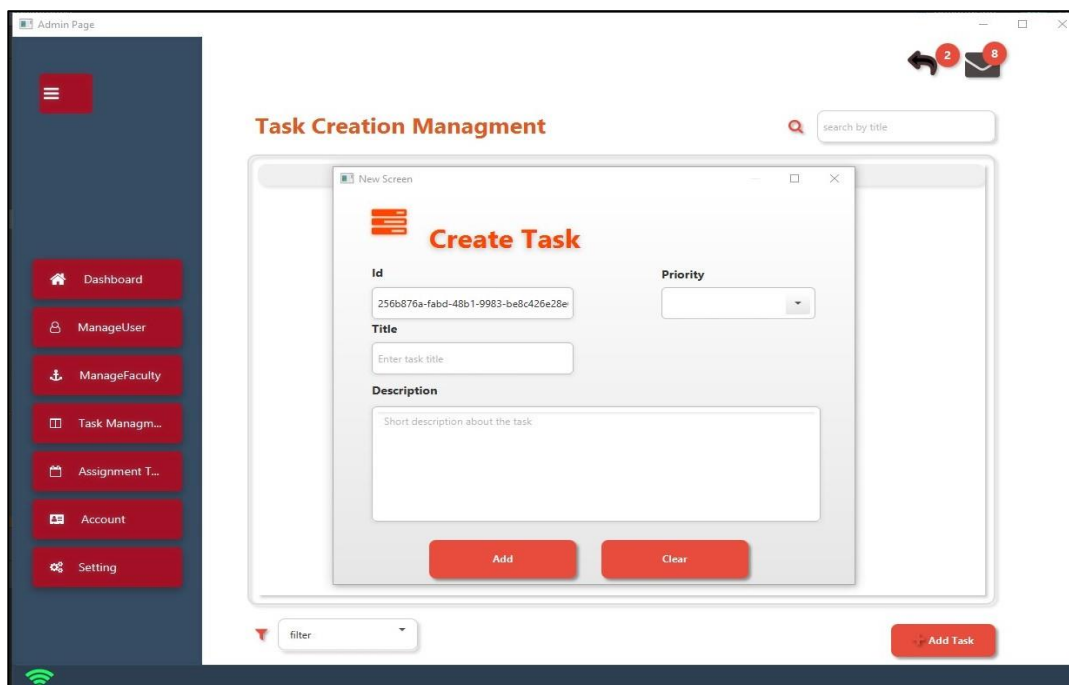


Figure 3.9 JavaFx Task creation page

- **Task Assign page:** In this page User can assign task to other users. Also in this page we are going to test the file operation while user may send many files at the same time. Figure 3.11 and Figure 3.12 shows the page for both frameworks.

Figure 3.10 windows forms Task assignment page

Figure 3.11 JavaFx Task assignment page

Chapter 4

Result

4.1 Introduction

In this part of the thesis, the results of evaluating and comparing the performance of two Frameworks , JavaFx and Windows Forms, which were used separately for the development of the online Task management system are presented. These results were obtained using various testing tools such as VisualVM, Windows Performance Analyzer and Visual Studio performance profiler in all performance factors, which are CPU usage, memory usage, application start up time, CRUD operation and application response Time.

4.2 File operation Read/Write

4.2.1 JavaFx

In Table 4.1 Javafx is tested by VisualVm.

Operation	Low	Average	Maximum
Read	47.12	48.71	50.3
Write	51.2	56	60.8

As you can see the maximum execution time for reading a file is 50.3ms with average of 48.71ms and for writing with maximum 60.8ms with average of 56ms.

4.2.2 Windows forms

In Table 4.2 windows forms is tested by Visual studio performance profiler.

Operation	Low	Average	Maximum
Read	33	34	35
Write	13	13.5	14

As you can see the maximum execution time for reading a file is 35ms with average of 34ms and for writing with maximum 14ms with average of 13.5ms.

4.3 Response time during user interactions

4.3.1 JavaFx

In Table 4.3 Javafx is tested by Visual Vm.

Factor	Low	Average	Maximum
Response Time	12.300	12.350	12.410

As you can see the maximum execution time for opening a window while click on a button is 12.410ms with average of 12.350ms.

4.3.2 Windows forms

In Table 4.4 Windows forms is tested by Visual studio performance profiler.

Factor	Low	Average	Maximum
Response Time	17	26.5	36

As you can see the maximum execution time for opening a window while click on a button is 36ms with average of 26.5ms.

4.4 CRUD operation with MySQL

4.4.1 JavaFx

In Table 4.5 Javafx is tested by VisualVm .

Operation	Low	Average	Maximum
Create	17.1	37.5	57.9
Read	0.530	1.1	1.67
Update	12.1	14.65	17.2
Delete	5.77	6.275	6.78

As you can see the maximum execution time for inserting data is 57.900ms with average of 37.500ms, read data with maximum 1.670ms with average of 1.100ms, update data with maximum 17.200ms with average of 14.650ms and delete data with maximum 6.780ms with average of 6.275.

4.4.2 Windows Forms

In Table 4.6 Windows form is tested by Visual studio performance profiler.

Operation	Low	Average	Maximum
Create	21	21.5	22
Read	0.117	0.165	0.213
Update	2	3	4
Delete	3.312	3.711	4.11

As you can see the maximum execution time for inserting data is 22ms with average of 21.5ms, read data with maximum 0.213ms with average of 0.165ms, update data with maximum 4ms with average of 3ms and delete data with maximum 4.11ms with average of 3.711

4.5 CRUD operation with Firebase

4.5.1 JavaFx

In Table 4.7 Javafx is tested by VisualVm .

Operation	Low	Average	Maximum
Create	10.38	11.27	12.16
Read	0.1634	0.1902	0.217
Update	27.571	26.892	26.213
Delete	0.1886	0.2158	0.243

As you can see the maximum execution time for inserting data is 12.160ms with average of 11.270ms, read data with maximum 0.217ms with average of 0.190ms, update data with maximum 26.213ms with average of 26.892ms and delete data with maximum 0.243ms with average of 0.215ms.

4.5.2 Windows Forms

In Table 4.8 Windows form is tested by Visual studio performance profiler.

Operation	Low	Average	Maximum
Create	33	36	39
Read	13	14.5	16
Update	34	38	42
Delete	0.53	0.67	0.81

As you can see the maximum execution time for inserting data is 39ms with average of 36ms, read data with maximum 16ms with average of 14.5ms, update data with maximum 42ms with average of 38ms and delete data with maximum 0.81ms with average of 0.67ms.

4.6 CPU Usage

4.6.1 JavaFx

In Table 4.9 Javafx is tested by Visual Vm.

Factor	Low	Average	Maximum
CPU	36.2	45.405	54.61

As you can see the maximum CPU usage while starting app is 54.61% with average of 45.405%.

4.6.2 Windows forms

In Table 4.10 Windows forms is tested by Visual studio performance profiler

Factor	Low	Average	Maximum
CPU	11	12.5	14

As you can see the maximum CPU usage while starting app is 14% with average of 12.5%.

4.7 Memory Usage

4.7.1 JavaFx

In Table 4.11 Javafx is tested by VisualVm.

Factor	Low	Average	Maximum
Memory	44.603	46.798	48.993

As you can see the maximum memory is 48.993MB with average of 46.798MB.

4.7.2 Windows forms

In Table 4.12 Windows forms is tested by Visual studio performance profiler

Factor	Low	Average	Maximum
Memory	49	48	50

As you can see the maximum memory is 50MB with average of 48MB.

4.8 Startup Time

4.8.1 JavaFx

In Table 4.13 Javafx is tested by Windows Performance Analyzer

Factor	Low	Average	Maximum
Time	13.125	15.430	17.736

As you can see the maximum time 17.736sec with average of 15.430sec.

4.8.2 Windows forms

In Table 4.14 Windows Forms is tested by Windows Performance Analyzer.

Factor	Low	Average	Maximum
Time	11.109	12.485	13.861

As you can see the maximum time 13.861sec with average of 12.485.

Chapter 5

Discussion

5.1 Introduction

Performance testing through discussion of the two Frameworks, namely JavaFx and Windows Forms, in developing online Task management systems is what this thesis aims for. The performance of these two Framework was evaluated based on four important factors such as are CPU usage, memory usage, application start up time, CRUD operation and application response Time, with the use of performance measurement tools VisualVM, Windows Performance Analyzer and Visual Studio performance profiler. In this chapter we will discuss about obtained result and also evaluate with previous available researches.

5.2 File Operation Read/Write

In Chapter 4, The Data result for JavaFx in Table 4-1 and Windows forms Table 4-2, comes from where we write 20 files per operation and the same for read the data from files. As discussed in Chapter 2, one of the performance comparison factors considered by Author Biswa et al. (2016) was file reading and displaying. In his study, which compared C, Fortran, C#, and Java, C# was found to perform the best (Biswa et al., 2016). Another research, provided by Chen (2010), compares C, C++, C#, and Java. A key factor for performance in this paper is represented by file operations. Once more, C# occupies the leading position (Chen, 2010). When observing the result, we notice that Windows Forms is ahead while performing a file read operation. For file writing operations, Windows Forms performs far better than JavaFX. To summarize, JavaFX performs a read operation on a file with an average speed of executions at 48.71ms, whereas Windows Forms performs it at 34ms. In the case of the writing of files, JavaFX does it at an average score of 56ms, while Windows Forms did better in execution with an average score of 13.5ms. This reflects that there is a big difference between the performance of this factor. In this factor, Windows Forms were better compared to JavaFX.

5.3 Response time

In this thesis response time refers to where a user clicks on a button and the framework send an event for execute that operation. So in this case according to Figure 3.9 and Figure 3.10 for both app, while user click on create button, a Creation task form is loaded, so the time that it takes to load that form and response to user is obtained for Javafx in Table 4.3 and for Windows forms in Table 4.4. We got an average time of 12.350ms in the case of JavaFX, whereas for Windows forms, we get the average time of 26.5ms, which means JavaFX UI response is twice as fast compared to Windows Forms. For instance, one of the reasons why JavaFX is better

performing than Windows Forms is that Author Adegeo (2023) writes Windows Forms uses GDI+—Graphics Device Interface, a software-based rendering engine responsible for all graphical tasks, including drawing, painting, and text rendering. It is a conventional solution for the implementation of UI but does not, by default, use hardware acceleration as it is in JavaFX and normally performs most of the activity with the help of the CPU. This makes it less effective in rendering modern animations and complex UI elements (Adegeo, 2023). On the other hand, JavaFX uses a modern hardware-accelerated rendering pipe, generally known as Prism. This pipeline makes use of hardware acceleration through GPU and takes the scene graph of JavaFX and the graphical elements for efficient rendering. These Prism capabilities together with Java 2D and Scene Graph make JavaFX appropriate for complex graphical tasks, animations, and fluent UI (JavaFX Overview, n.d.). Therefore, based on this, our results also depict that JavaFX is way more productive and faster in terms of comparison with Windows Forms.

5.4 CRUD Operation with Databases

For a better performance comparison, in this thesis, I have compared performance using both local and server-side (Firebase) databases. In this experiment, the operations performed were adding, editing, deleting, and updating a user 30 times in both databases to obtain more accurate results.

5.4.1 MySQL CRUD operation

In Chapter 4, the data results come for JavaFX in Table 4.5 and Windows Forms in Table 4.6, we achieved the following results:

- **Insert:** JavaFX did that in 37.5ms on average while Windows Forms did the same in 21.5ms, which is faster.
- **Read:** JavaFX did it in 1.1ms on average, while Windows Forms performed the same operation in 0.165ms, which is faster.
- **Update:** JavaFX executed this with an average time of 14.65ms, which is well outperformed by Windows Forms performing the same operation in 3ms.
- **Delete** JavaFX did it at an average of 6.275ms, whereas Windows Forms did the same in 3.711ms, which is faster.

5.4.2 Firebase CRUD operation

In Chapter 4, the data results for JavaFX in Table 4.7 and Windows Forms in Table 4.8, we achieved the following results:

- **Insert:** JavaFX did that in 11.27ms on average while Windows Forms did the same in 36ms, which goes to show how faster JavaFX is.
- **Read** JavaFX executed this with the average time of 0.190ms, while Windows Forms executed for 14.5ms, proving JavaFX the fastest.
- **Update:** This was done by JavaFX in an average time of 26.892ms against Windows Forms, which did the same operation at approximately 38ms. Thus, JavaFX outpaces it.
- **Delete:** This JavaFX executed with an average of 0.2158ms, while doing this very same thing, Windows Forms took up to 0.67ms, which says JavaFX is much quicker.

Unfortunately, no prior research is available with which we can cross-compare our findings; thus, we have to rely on our results. We deduce from these results that Windows Forms are better with MySQL, whereas JavaFX is better with Firebase for online operations.

5.5 CPU usage

In the thesis, we have tested the CPU usage to find out how much CPU is the application munching during the time of its startup. This was done many times in order to get an exact result. Results are given in Table 4.9 for JavaFX and Table 4.10 for Windows Forms, with averages of 45.405% CPU usage by JavaFX and 12.5% CPU usage by Windows Forms. Unfortunately, there is no prior research with which to compare our findings; we must go by our own results. Based on these results, it would appear that Windows Forms performed more efficiently than JavaFX.

5.6 memory Usage

In this thesis, we measured the memory consumption of the applications to understand how much memory each application used upon initialization. We repeated the test many times in order to be as exact as possible. Table 4.11 and 4.12 show that JavaFX uses approximately 46.798 MB of memory while Windows Forms consumes approximately 48 MB of memory at

the start. Although JavaFX seems to use slightly better memory, this small difference is unable to support the idea that Windows Forms are significantly less efficient than JavaFX. Unfortunately, there does not appear to be any comparative research available, and we must support our findings from the experiment and conclude that indeed JavaFX runs slightly better.

5.7 Startup time

For the purposes of this thesis, startup time will be defined as the instant when the user clicks the application to start it. When the application is clicked, it needs to load the most recent data from the server and load resources necessary for starting the application. Results presented in Table 4.13 show JavaFX, while Table 4.14 presents Windows Forms; the average time it takes for starting the app is 15.430 seconds by using JavaFX against 12.485 seconds by using Windows Forms—hence, much faster than JavaFX.

5.8 Summary

In the comparison we made between JavaFX and Windows Forms, it can be depicted that Windows Forms is more efficient as compared to JavaFX in CPU usage, startup time, MySQL CRUD operation, and file operation. On the other hand, JavaFX performs better in response time, Firebase CRUD operation, and a little difference in memory utilization where JavaFX performs better as compared to Windows Forms.

Chapter 6

Conclusion

6.1 Introduction

The conclusion of this chapter brings about the performance comparison between JavaFX and Windows Forms with regard to research answers posed by the thesis. Additionally, it explains certain limitations encountered in conducting this study and goes further to make recommendations to any future researcher who may wish to undertake a scientific investigation into the aspects of JavaFX and Windows Forms.

6.2 Conclusion

This work intends to compare the performances of two frameworks: JavaFX and Windows Forms using the .NET Framework. In this work, all results are based on measurements taken in six categories: CPU usage, memory usage, application start-up time, MySQL CRUD operations, Firebase CRUD operations, and application response time.

Overall, Windows Forms were much better compared to JavaFX in the case of file operations like reading and writing. Also, JavaFX was taking less response time in every user interaction, while Windows Forms were a bit slower. JavaFX was also fast in CRUD operations with Firebase, but somehow, Windows Forms did not perform as expected. However, for operations with MySQL, Windows Forms were somewhat better, while JavaFX was slower.

JavaFX and Windows Forms used almost the same amount of memory; both frameworks managed the given workload efficiently, but JavaFX did it a bit better. The CPU was used more by JavaFX compared to Windows Forms, where on Windows Forms, it used four times less, which is huge. The time at startup also differed a little, with Windows Forms starting a bit faster than JavaFX.

The results therefore show that, on memory management and response time, JavaFX outperformed, and CRUD operation with Firebase, but for Windows Forms, it performed better on CPU usage, CRUD operations with MySQL, application startup time, and file operations, doing quite well on CPU usage and file handling.

Table 6.1 shows the summary result for both frame work.

Performance comparison summary table		
Performance factor	JavaFx	Windows forms
File operation		best
Response time	best	
MySQL CRUD		better
Firebase CRUD	better	
CPU usage		best
Memory Usage	slightly better	
Startup time		better

6.3 Limitation

Even though this thesis contributed much to provide valuable results about the performance comparison of two popular frameworks, JavaFX and Windows Forms, a number of limitations have been faced. Firstly, even though Windows was used as the operating system for testing the research results, other operating systems like Linux or macOS could show different outcomes. Besides this, during the last three years, Windows Forms have not received any significant feature updates from Microsoft, while JavaFX is regularly updated. This could mean that, as time progresses, the findings of this study may have a chance of becoming outdated. Moreover, we found few research and papers to contextualize our findings with; thus, it is quite difficult to contextualize our findings with the greater academic debates. Also, in this research, the dataset used was very small. For larger ones, which take a longer period of time to process, results may differ. The performance analysis was limited to only five key focuses: CPU usage and memory usage, application startup time, CRUD operations, and response time of applications. These, though being all-important, are not exhaustive; because of the constraints of time, this study focused chiefly on these factors. Finally, this study has only researched some of the limited features concerning memory and CPU usage. There is a need for testing in various parts of the application to present comprehensive results.

6.4 Suggestion

The performance comparison of JavaFX versus Windows Forms issued a number of significant differences in their performance, richness of UI, and resource efficiency. While Window Forms showed good results in local CRUD operations, CPU usage, startup time, and file handling, it lacked the delivery of modern UI designs and rich multimedia features whereby JavaFX outperformed it by being able to respond twice faster than Windows Forms. Based on these findings, recommendations include the following:

For Students: JavaFX is recommended for those projects which require a rich, creative, and visually appealing graphical user interface targeted at animations, video-type presentations, and modern design. Its advanced graphical and media support makes it good to go for complex, bigger-sized projects, while Windows Forms would be more suitable in simple projects where all the attention would be towards mere functionality rather than appeal. It provides a much more resource-friendly alternative and works perfectly for applications where UI complexity is not of primary importance.

For Project Managers and Developers: While considering the choice between JavaFX and Windows Forms for commercial applications, the following aspects need to be considered: Windows Forms is the better option for those applications in which file processing is the major task and UI complexity is not significant. With it, there is faster development time and greater performance with huge volumes of files, while JavaFX should be selected when the application requires a more modern, rich user experience. This will generally be more suitable for projects in which serving advanced user interactions and modern design elements is paramount, as it can support complex graphic interfaces, multimedia, and scalable UI designs.

For Future Research: These areas of study are recommended to the researcher who is willing to really flesh out these frameworks:

- **Explore WPF (Windows Presentation Foundation):** A comparison of JavaFX to more modern alternative replacements for Windows Forms would provide more detail than simple UI designs and performance tests.
- **Test on Multiple Operating Systems:** Performance testing on Linux and macOS, in addition to Windows, would allow a broader understanding of the performance these frameworks have in said environments.

- **Work with Larger Datasets:** Testing with larger datasets will give a much clearer picture about the scalability and performance of these frameworks on complex tasks.
- **Examine Additional Performance Metrics:** Some other factors which are to be kept in mind include disk speed, network latency, and response time under different loads for a comprehensive performance evaluation.
- **Analyze Memory and CPU Usage:** Performing an in-depth analysis of the memory and CPU usage in performing various operations, such as CRUD activities, response time, and startup time, may reveal important facts about resource management.

References

- Adegeo. (2023, February 7). Overview - Windows Forms .NET Framework. Microsoft Learn. <https://learn.microsoft.com/en-us/dotnet/desktop/winforms/windows-forms-overview?view=netframeworkdesktop-4.8>
- Biswa, K., Jamatia, B., Choudhury, D., & Borah, P. (2016). Comparative analysis of C, FORTRAN, C# and Java programming languages. *International Journal of Computer Science and Information Technology*, 7(2), 1004-1007.
- Chen, H. (2010). Comparative study of C, C++, C#, and Java programming languages.
- Cheshta, & Sharma, D. (2017). JavaFX framework and comparative analysis. *International Journal of Computer Science & Communication (IJCSC)*, 8(2), 1-4. Available at: <http://csjournals.com/IJCSC/PDF8-2/1.%20Chestha.pdf>
- Deitel, H., & Deitel, P. (2013). *Java how to program* (9th ed.). Pearson Education Limited.
- Deitel, P., & Deitel, H. (2016). *Visual C# how to program* (6th ed.). Pearson. <https://www.amazon.com/dp/0134601548>
- Gewarren. (2023, March 30). Overview of .NET Framework - .NET Framework. Microsoft Learn. <https://learn.microsoft.com/en-us/dotnet/framework/get-started/overview>
- Gewarren. (2024, January 10). Introduction to .NET - .NET. Microsoft Learn. <https://learn.microsoft.com/en-us/dotnet/core/introduction>
- Hamaamin, R. A., Ali, O. M. A., & Kareem, S. W. (2024). Java programming language: Time permanence comparison with other languages: A review. In *ITM Web of Conferences* (Vol. 64, p. 01012). EDP Sciences.
- Heger, C., van Hoorn, A., Mann, M., & Okanovic, D. (2017, April 22-26). Application performance management: State of the art and challenges for the future. In *Proceedings of the 8th ACM/SPEC International Conference on Performance Engineering (ICPE 2017)* (pp. 123-130). Association for Computing Machinery.
- Hong, J., & Ryu, S. (2023, February 23). Introduction to programming languages.

JavaFX Overview (Release 8). (n.d.). <https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm>

Laying out a user interface with JavaFX 2.0. (n.d.). Oracle. <https://www.oracle.com/technical-resources/articles/java/layoutfx.html>

Loeffelmann, K. (2022, January 14). State of the Windows Forms designer for .NET applications - .NET blog. *The .NET Blog*. <https://devblogs.microsoft.com/dotnet/state-of-the-windows-forms-designer-for-net-applications/>

Pajjuri, A., & Ahmed, H. (2001). A performance analysis of Java and C. University of Columbia.

Sharma, S. (2019). Performance comparison of Java and C++ when sorting integers and writing/reading files